АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ В МИКРОЭЛЕКТРОНИКЕ.
ТЕОРИЯ, МЕТОДЫ, АЛГОРИТМЫ
(Вычислительные системы)
1978 год Выпуск 77

УДК 519.17

ПРИМЕНЕНИЕ ОТНОСИТЕЛЬНЫХ РАЗБИЕНИЙ ДЛЯ ПОИСКА КЛИК

Ю.Е.Бессонов, В.А.Скоробогатов

Пусть $G=(V, E)$ – конечный неориентированный граф без петель и кратных ребер, $|V| = p$, $|E| = q$. Кликой называют любой максимальный (по включению) полный подграф в G.

Для некоторой вершины u рассмотрим относительное разбиение [1,2] $\hat{V}(u)$ множества вершин графа G:

$$\hat{V}(u) = \{V_i \mid V_i \subseteq V,\ v \in V_i \Leftrightarrow d(v,u) = i,\ 0 \le i \le k(u)\},$$

где $d(v, u)$ – расстояние между вершинами v и u, а $V_0 = \{u\}$. Подграф, образованный любой парой соседних слоев (V_j, V_{j+1}), $0 \le j \le k(u) - 1$, будем называть поясом.

Пусть $\mathcal{L}_0 = G$, а $\mathcal{L}_1 = \{L_1, L_2, \ldots, L_{k(u)}\}$ – множество поясов. Для каждого $L_j \in \mathcal{L}_1$ выберем $u^* \in V_j \cup V_{j+1} \equiv V_j$ с минимальной степенью в L_j и построим разбиение $\hat{V}_j^1(u^*) = \{V_{ji}^1 \mid 0 \le i \le k(u^*)\}$. Получим множество $\mathcal{L}_2$, каждый элемент которого – пояс, образованный парой $(V_{ji}^1, V_{j,i+1}^1)$, $0 \le i \le k(u^*)-1$, $0 \le j \le k(u)-1$. Пусть $\mathcal{L}_s$ $(s \ge 1)$ – множество поясов, полученных применением описанной выше процедуры к $\mathcal{L}_{s-1}$. Тогда имеет место следующее

УТВЕРЖДЕНИЕ. При некотором h каждый подграф из $\mathcal{L}_h$ является либо кликой в G, либо частью некоторой клики.

Действительно, поскольку полный подграф имеет диаметр 1, то при любом s он целиком содержится в некотором $L \in \mathcal{L}_s$, а с увеличением s порядок подграфов из $\mathcal{L}_s$ уменьшается. Выбор вершины с минимальной степенью в L обеспечивает разбиение множества вершин подграфа L на число слоев $k(u) \ge 2$, причем равенство достигается тогда и только тогда, когда L – полный подграф. (В самом деле, пусть

L – неполный подграф, u^* – вершина с минимальной степенью и $\hat{V}(u^*) = \{\{u^*\}, V_1\}$. Так как L неполный, то существует $u \in V_1$ такая, что $\deg u < \deg u^*$.)

Наименьшее h, при котором $\mathcal{L}_h$ содержит только полные подграфы, назовем глубиной алгоритма. Если из $\mathcal{L}_h$ выписать (без повторений) все максимальные полные подграфы, то получим набор всех клик графа G. Если интерес представляют только k-клики, то необходимо ограничиться таким наименьшим $h(k)$, при котором каждый подграф из $\mathcal{L}_{h(k)}$ либо полный, либо имеет число вершин меньше k, либо имеет число ребер меньше $k\cdot(k-1)/2$.

Перечисление клик удобно описывать в терминах дерева поиска $T(G)$. Корневая вершина $T(G)$ соответствует исходному графу G. Вершины $T(G)$, смежные с корневой, соответствуют подграфам $L_j \in \mathcal{L}_1$ $(j = 1,2,\ldots,k(u))$. Далее, по индукции, пусть $w_1^i,\ldots,w_{n(i)}^i$ – множества вершин $T(G)$, находящихся на расстоянии i от корневой $(i \geq 1)$, соответствующих подграфам $L_j^i = (V_j^i, E^i)$, $E^i \subseteq V_j^i \times V_j^i$, $1 \leq j \leq n(i)$. Тогда если L_j^i – полный подграф, то вершина w_j^i висячая. В противном случае из нее выходят ветви к вершинам $w_1^{i+1},\ldots,w_{n(j)}^{i+1}$, соответствующим подграфам $L_1^{i+1},\ldots,L_{n(j)}^{i+1}$, которые образованы всеми поясами в относительном разбиении $\hat{V}_j^i(u^*)$ множества вершин подграфа L_j^i. Может оказаться, что пояс L_j^i несвязный. В этом случае мы будем считать, что каждой компоненте пояса L_j^i соответствует подмножество вершин из $\{w_1^{i+1},\ldots,w_{n(j)}^{i+1}\}$.

§1. Описание алгоритмов

Для простоты описания используем стеки [3] S_1, S_2, S_3. В S_1 записываются пары (V_i^*, V_{i+1}^*), соответствующие поясам в относительном разбиении некоторого графа G^*, либо множества вида $\{u\}$, если G^* вполне несвязен. Элементами S_2 будут числа, а S_3 будет содержать на каждом шаге набор подмножеств $V_1,\ldots,V_s (V_i \subset V)$, обладающий тем свойством, что каждая вновь найденная клика, не содержащаяся целиком в каждом $V_i \in S_3$, $i = 1,2,\ldots,s$, не была получена на предыдущих шагах.

Введем следующую символику. Если значение x помещается на верх стека S, то будем писать $S \Leftarrow x$. Запись $x \Leftarrow S$ будет означать, что переменная x принимает значение верхнего элемента S и этот элемент исключается из S. Верхний элемент S обозначим через $top(S)$. Текущие значения числа вершин и ребер подграфов обозначены соответственно через p_i и q_i.

АЛГОРИТМ А1 (перечисление клик).

1^0. $p_t := p$; $q_t := q$; $V^* := V$; $E^* := E$.

2^0. а) В графе $G^* = (V^*, E^*)$ выбираем u^* такую, что $\deg u^* = \min_{u \in V^*} \{\deg u\}$, и строим $\hat{V}^*(u^*) = \{V^*_i \mid 0 \le i \le k(u^*)\}$;

б) записываем в S_1 в обратном порядке все пары (V^*_i, V^*_{i-1}), $i = k(u^*), k(u^*) - 1, \ldots, 1$;

в) помечаем все вершины из V, вошедшие в разбиение;

г) $S_2 \Leftarrow k(u^*)$; $S_3 \Leftarrow \{0\}$.

3^0. Если $\mathrm{top}(S_2) = 0$, то $\mathrm{top}(S_2) \Leftarrow S_2$, $\mathrm{top}(S_3) \Leftarrow S_3$, и повторяем выполнение п.3^0; если $S_2 = \emptyset$ или $\mathrm{top}(S_2) \neq 0$, то переходим к п.4^0.

4^0. Если $S_1 = \emptyset$, то идем к п.8^0, иначе:

а) $(V^*_i, V^*_{i-1}) \Leftarrow S_1$;

б) снимаем метки со всех вершин этой пары;

в) $\mathrm{top}(S_2)$ уменьшаем на единицу;

г) $\mathrm{top}(S_3) \Leftarrow S_3$, $S_3 \Leftarrow V^*_{i-1}$;

д) $V^* := V^*_i \cup V^*_{i-1}$; $G^* := (V^*, E^*)$, где $E^* = \{(u,v) \mid (u, v) \in E \wedge u \in V^* \wedge v \in V^*\}$;

е) $p_t := |V^*|$, $q_t := |E^*|$.

5^0. Если $q_t \neq p_t (p_t - 1)/2$, то идем к п.2^0. Иначе:

6^0. а) проверяем полный подграф на максимальность

$$\bigcap_{u \in V^*} \{v \in V \mid (u,v) \in E\} = \emptyset.$$

Если пересечение непусто, то переходим к п.7^0, иначе:

б) если V^* целиком не содержится ни в каком элементе S_3, то выводим клику и переходим к п.7^0; иначе переходим к п.7^0.

7^0. Помечаем все вершины из V^*. Строим V' – множество всех непомеченных вершин из V. Если $V' = \emptyset$, то переходим к п.3^0. Иначе $V^* := V'$, и переходим к п.2^0.

8^0. Конец.

АЛГОРИТМ А2 (поиск k-клик в графе) отличается от предыдущего только п. 5^0.

5^0. а) Если $p_t < k$, то идем к п.7^0, иначе проверяем:

б) если $q_t < k(k-1)/2$, то идем к п.7^0, иначе:

в) если $q_t \neq p_t (p_t - 1)/2$, то идем к п.2^0, иначе:

г) если $p_t \neq k$, то идем к п.7^0, иначе идем к п.6^0.

В следующем алгоритме через R будем обозначать множество, элементами которого являются пары смежных слоев вершин.

АЛГОРИТМ А3 (поиск максимальной клики).

I^0. $p_t := p$; $q_t := q$; $V^* := V$; $E^* := E$; $G^* := (V^*, E^*)$ $R := \emptyset$.

2^0. а) В графе G^* выбираем u^* такую, что $\deg u^* = \min_{u \in V^*} (\deg u)$, и строим $\hat{V}^*(u^*)$;

б) $R := R \cup \{V_i^* u V_{i+1}^*\}_{i=0}^{k(u^*)-1}$;

3^0. Строим V^* – множество непомеченных вершин из V. Если $V^* = \emptyset$, то идем к п.4^0. Иначе – к п.2^0.

4^0. а) Удаляем из R множество V' с максимальной мощностью и снимаем пометки со всех вершин V';

б) $V^* := V'$; $G^* := (V^*, E^*)$. . Если G^* неполный, то переходим к п.2^0. Иначе:

5^0. Выводим клику. Конец.

§2. Верхние оценки сложности вычислений

2.1. О ц е н к и ч и с л а о п е р а ц и й. Введем следующее

ОПРЕДЕЛЕНИЕ. Вершину графа G, смежную со всеми вершинами некоторого подграфа, будем называть центром этого подграфа.

Для установления верхних оценок будем считать, что при выборе начальной вершины для относительного разбиения не требуется минимальности ее степени. Пусть начальная вершина будет произвольной, не являющейся центром. (В самом деле, выбор вершины с минимальной степенью – эвристический прием, ускоряющий во многих случаях сходимость алгоритма, однако существуют примеры, в которых при выборе вершин с минимальной степенью алгоритм работает дольше, чем с иными правилами выбора.)

Пусть $i \geq j \geq 0$ – целые числа. Определим по индукции двоичное дерево T_{ij} следующим образом. Паре (i,j) сопоставим корневую вершину. Если некоторой вершине $w \in T_{ij}$ приписана пара (b,c), то а) при $b = c$ или $b = c+1$ w висячая; б) при $b \geq c+2$ из w выходят последователи, которым приписываются пары $(b-1, c+1)$ и $(b-1, c)$. Очевидно, что для каждой пары (i, j), где $i \geq j \geq 0$, дерево T_{ij} строится однозначно.

Пусть $\phi(i,j)$ обозначает число вершин дерева T_{ij}.

ЛЕММА. Ф у н к ц и я $\phi(i,j)$ у д о в л е т в о р я е т р е к у р р е н т н ы м с о о т н о ш е н и я м:

$$\phi(i,j) = \phi(i-1,j+1) + \phi(i-1,j) + 1, \quad i \geq j+2,$$
$$\phi(i,0) = \phi(i-1,0) + \phi(i-2,0) + 1, \quad i \geq 2.$$

Лемма легко доказывается по индукции.

ТЕОРЕМА I. Для любого $G \in \mathfrak{G}(p)$, где $\mathfrak{G}(p)$ - класс всех графов порядка p, число вершин дерева $T(G)$ не превосходит числа вершин дерева $T_{p,0}$.

ДОКАЗАТЕЛЬСТВО будем вести индукцией по числу p. Для $p = 1,2$ теорема верна. Пусть теорема верна для любого $G \in \mathfrak{G}(k)$, где $k \le p-1$. Обозначим через $|T(G)|$ число вершин дерева $T(G)$. Тогда

$$\max_{G\in\mathfrak{G}(p)} |T(G)| = \max_{G\in\mathfrak{G}(p)} (1 + \sum_{i=1}^{r} |T(G_i)|),$$

где G_i - пояса в некотором относительном разбиении графа G. По предположению индукции,

$$\max_{G\in\mathfrak{G}(p)} (1 + \sum_{i=1}^{r} |T(G_i)|) =$$

$$= \max_{R\in\mathcal{R}} (1 + \phi(p_1 + 1,1) + \sum_{i=1}^{r-1} \phi(p_i+p_{i+1},0)), \qquad (I)$$

где p_i - число вершин в i-м слое, $\mathcal{R}$ - множество всех упорядоченных разбиений целого числа p на части, $\mathcal{R} = \{ R|R=(1,p_1,\dots, p_i); p_i \ge 1; i = \overline{1,r}; \sum_{i=1}^{r} p_i = p-1; r=2,3,\dots,p-1\}$. Очевидно, $\mathcal{R} = \bigcup_{j=1}^{p-1} \mathcal{R}_j$, где

$$\mathcal{R}_j = \{R|R=(1,p_1,\dots,p_j); p_s \ge 1; \ s = \overline{1,j}; \ \sum_{s=1}^{j} p_s = p-1\}.$$

Поэтому

$$\max_{G\in\mathfrak{G}(p)} |T(G)| = \max_{2\le j\le p-1} \{\max_{\mathcal{R}_j} F(1,p_1,\dots,p_j)\},$$

где $F(1,p_1,\dots,p_j) \equiv 1+\phi(p_1+1,1) + \sum_{i=1}^{r-1} \phi(p_i+p_{i+1},0)$.

С помощью леммы можно установить следующие соотношения:

$$\max_{\mathcal{R}_j} F(1,p_1,\dots,p_j) = F(\underbrace{1,1,\dots,1}_{s},p-j,1,\dots,1), \qquad (2)$$

$$\max_{2\le j\le p-1} F(\underbrace{1,1,\dots,1}_{s},p-j,1,\dots,1) = F(1,p-2,1), \qquad (3)$$

где $s \ge 1$.

Доказательство этих соотношений несложно, поэтому мы его опускаем. Тогда из (I)-(3) следует

$$\max_{G} \left(1 + \sum_{i=1}^{r} |T(G_i)|\right) = 1 + \phi(p-1,1) + \phi(p-1,0) = \phi(p,0).$$

Теорема доказана.

ЗАМЕЧАНИЕ. До сих пор мы не касались вопроса о существовании таких графов G, для которых $T(G) = T_{po}$. Легко показать, что такими графами являются, в частности, $\overline{P}_p$ – дополнения простой цепи.

ТЕОРЕМА 2. Число операций при поиске всех клик в графе $G \in \mathcal{Y}(p)$ имеет в худшем случае порядок $O(p^2(1{,}62)^p)$.

ДОКАЗАТЕЛЬСТВО. Подсчитаем число вершин дерева T_{po}. Для этого найдем все члены последовательности $\phi(p,0)$. Пусть, по определению, $\phi(0,0) = 1$. Тогда $\phi(1,0) = 1$, а для $p \geq 2$ согласно лемме $\phi(p,0) = \phi(p-1,0) + \phi(p-2,0) + 1$. Пользуясь методом производящих функций [4], найдем

$$\phi(p,0) = \lambda\left(\frac{2}{\sqrt{5}-1}\right)^{p+1} - \mu\left(\frac{2}{-1-\sqrt{5}}\right)^{p+1} - \lambda\left(\frac{2}{\sqrt{5}-1}\right)^{p} + \mu\left(\frac{2}{-1-\sqrt{5}}\right)^{p} +$$

$$+ \lambda\left(\frac{2}{\sqrt{5}-1}\right)^{p-1} - \mu\left(\frac{2}{-1-\sqrt{5}}\right)^{p-1} - 1; \quad \lambda = \frac{2}{5(3-\sqrt{5})}, \quad \mu = \frac{2}{5(3+\sqrt{5})}, \quad p \geq 2.$$

Отсюда видно, что $\phi(p,0) = O\left(\left(\frac{2}{\sqrt{5}-1}\right)^p\right) = O((1{,}62)^p)$.

Из построения алгоритма AI видно, что с каждой вершиной w дерева $T(G)$ связаны операции относительного разбиения и подсчета ребер (когда w не висячая) и подсчета ребер, проверки полного подграфа на максимальность и отсутствие дубликатов (когда w висячая). Эти операции в худшем случае выполняются за время $O(p^2)$.

Поэтому для графов с p вершинами время работы алгоритма в худшем случае имеет порядок $O(p^2(1{,}62)^p)$. Теорема доказана.

Интересно отметить, что графы, на которых время работы алгоритма AI достигает своего максимального значения, не являются клико-экстремальными. Основное время в этом случае уходит на бесполезное выделение частей клик. На графах же Муна и Мозера [5] число вершин дерева $T(G)$ имеет порядок $O(3^{p/3})$. Эта величина растет асимптотически медленнее, чем $(1{,}62)^p$.

Это замечание касается также алгоритмов поиска всех клик (см. [6]), которые работают подобно нашему как на графах $\overline{P}_p$, так и на полных k-дольных графах.

Пусть теперь алгоритм AI модифицирован таким образом, что исходное разбиение множества вершин графа G диаметральное, т.е. $\hat{V}(u^*) = \{\{u^*\}, V_1, \ldots, V_{d(G)}\}$, где $d(G)$ – диаметр графа G.

Тогда из равенства (2) вытекает

$$|T(G)| \le F(1,1,\ldots,1,p-d(G),1,\ldots,1) = O((1{,}62)^{p-d(G)}) .$$

Следовательно, для времени работы τ_G модифицированного алгоритма AI можно записать:

$$\max_G \tau_G = O(p^2 (1{,}62)^{p-d(G)}) + O(p^3),$$

$$\max_G \tau_G = O(p^2 \sum_{i=1}^{d(G)} (1{,}62)^{p_{i-1}+p_i}) + O(p^3),$$

где $p_0 = 1, (p_1, p_2, \ldots, p_{d(G)})$ – диаметральная строка λ-матрицы [2], и, кроме того, считается, что на поиск диаметральной вершины требуется $O(p^3)$ операций.

Из последних соотношений видно, что для графов с малым числом ребер и большим диаметром предварительный анализ метрических характеристик позволяет уменьшать время работы алгоритма.

2.2. Оценки объема памяти.

ТЕОРЕМА 3. Количество ячеек памяти, необходимое для работы алгоритмов AI и A2, равно $O(p^2)$. Количество ячеек, необходимых для реализации алгоритма A3, имеет порядок $O(p(1{,}62)^p)$.

ДОКАЗАТЕЛЬСТВО. Количество ячеек, необходимых для работы алгоритмов AI и A2, можно выразить как

$$\Pi = \sum_{i=1}^{5} c_i \Pi_i + c_6,$$

где Π_1, Π_2, Π_3 – количество ячеек, занимаемых соответственно стеками S_1, S_2 и S_3; Π_4 – память, необходимая для хранения начальных данных, и Π_5 – для промежуточных результатов; c_i $(i = \overline{1,6})$ – некоторые константы.

В общем случае $\Pi_4 = O(p^2)$, $\Pi_5 = O(p)$. Припишем каждой вершине w дерева $T(G)$ вес $P(w)$, равный числу вершин в подграфе, соответствующем w. Тогда из построения алгоритмов AI и A2 следует

$$\Pi_1 \le \sum_{s=1}^{h} p(w_s), \quad \Pi_3 \le \sum_{s=1}^{h} p(w_s), \quad \Pi_2 \le h,$$

где h – глубина алгоритма и $w_1, w_2, \ldots, w_h$ – вершины $T(G)$, принадлежащие некоторой цепи, соединяющей корневую вершину $T(G)$ с вершиной на уровне h.

Так как $P(w_s) = O(p)$, $h = O(p)$, то $\Pi_1 = O(p^2)$, $\Pi_2 = O(p)$, $\Pi_3 = O(p^2)$. Поэтому $\Pi = O(p^2)$.

В алгоритме А3 на каждом шаге, вообще говоря, необходимо просматривать все ранее построенные пары смежных слоев. Следовательно, объем памяти должен иметь (в общем случае) порядок мощности множества вершин дерева $T(G)$. Поэтому для алгоритма А3 число ячеек не может расти быстрее $O(p \cdot (1{,}62)^p)$.

§3. Результаты эксперимента

Применялась статистическая оценка вычислительной сложности алгоритмов А1 и А2. Генератор случайных графов обеспечивал равновероятную выборку графов из класса $\mathfrak{G}(p,q)$ всех графов с p вершинами и q ребрами. Статистические характеристики алгоритмов получались в результате обработки данных путем реализаций алгоритмов при решении выбранных примеров. Объем выборки принимался равным 100.

В качестве характеристик вычислительной сложности алгоритмов были взяты а) число вершин дерева, б) число операций сравнения, связанных с просмотром ребер при поиске клик.

Программы написаны на языке ФОРТРАН ЭВМ "Минск-32" и позволяют обрабатывать графы с числом вершин $p \le 50$.

На рис.1 показаны зависимости числа операций $\varphi(p,q)$ при поиске всех клик в графах с числом вершин $p = 10, 15$ и 20 от числа ребер q.

На рис.2 приводятся аналогичные зависимости для числа вершин дерева $T(G)$.

Кроме того, получены зависимости для функции от числа p: $\max_q \varphi(p,q)$, $\varphi(p,2p)$, $\varphi(p,4p)$, где p меняется в пределах от 10 до 50 с шагом 5. Графики этих зависимостей показаны на рис.3.

Реальное время работы алгоритма в сек. на ЭВМ "Минск-32") для графа G с p вершинами и q ребрами можно выразить как $\tau_G(p,q) \approx 4{,}8 \cdot 10^{-4}\, \varphi(p,q) + 3{,}6$. В частности, для $p = 25$ имеем $\max_q \tau_G(25,q) \approx 302{,}7$ сек.

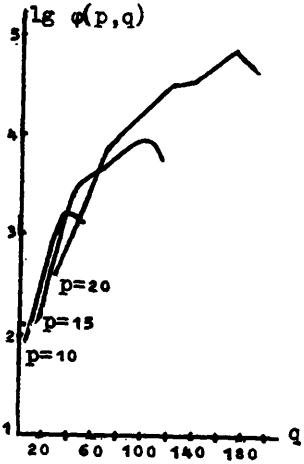

Рис. I

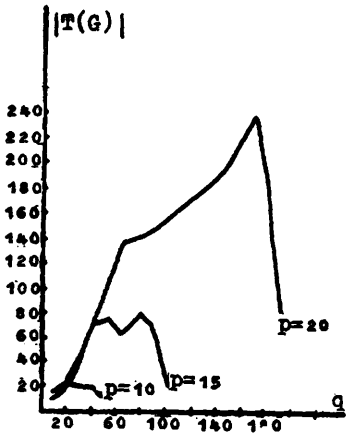

Рис. 2

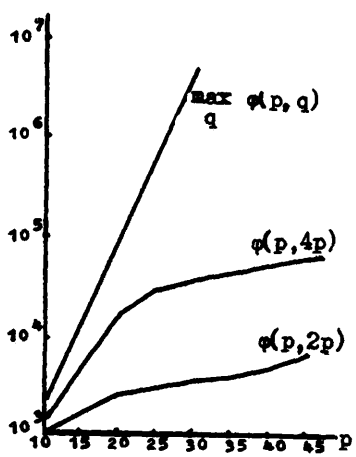

Рис. 3

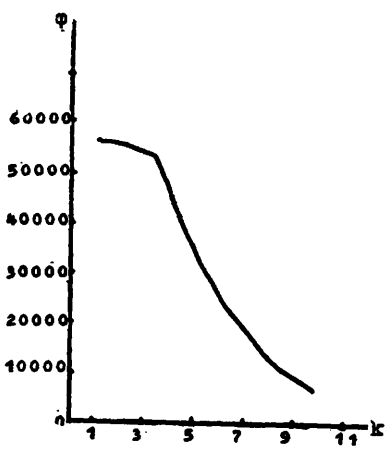

Рис. 4

На рис.4 изображен график зависимости числа просмотров ребер при поиске всех клик порядка k от величины k. Этот график получен для случайного графа с 30 вершинами и 148 ребрами.

Литература

I. СКОРОБОГАТОВ В.А. Анализ связности неографов. -В кн.: Автоматизация проектирования в микроэлектронике. Теория. Методы. Алгоритмы. (Вычислительные системы, вып. 64.) Новосибирск, 1975, с. 11-25.

2. СКОРОБОГАТОВ В.А. Относительные разбиения и слои графов. -В кн.: Вопросы обработки информации при проектировании систем. (Вычислительные системы, вып.69) Новосибирск, 1977, с.3-10.

3. КНУТ Д. Искусство программирования для ЭВМ. Т.I. Основные алгоритмы. М., "Мир", 1976.

4. ХОЛЛ М. Комбинаторика. М., "Мир", 1970.

5. MOON J., MOSER L. On cliques in graphs. - "Israel J.Math", 1965,v.3, N 1, p.23-28.

6. BRON C., KERBOSCH J. Finding all cliques of an undirected graph. - "Communs ACM", 1973,v.16,N 9, p.575-577.

Поступила в ред.-изд.отд.
15 апреля 1978 года